Future Directions in SGML/XML

Charles F. Goldfarb

Abstract

SGML is unique among information representations for its ability to preserve the abstract data content of arbitrarily complex document structures. That capability has led, over the past ten years, to the near-universal adoption of SGML for large-scale document processing. Now XML – the streamlined subset of SGML for the World Wide Web – is applying generalized markup to a vastly different class of applications, ranging from data interchange to self-describing software.

In this paper, Dr. Goldfarb, inventor of SGML and co-author of *The XML Handbook* (Goldfarb/Prescod 1998), explains the profound implications of this major shift in computing technology (as an *InfoWorld* editorial called it). He describes how the SGML family of International Standards has already been revised to accommodate these new applications, and other changes that can be expected in the near future.

Copyright ©1998 Charles F. Goldfarb. All rights reserved.

1 SGML, HTML, and XML

SGML: most widely-used but little-known

SGML is one of the world's most widely-used but little-known technologies. It is no exaggeration to say that our most complex and important social and technological systems are utterly dependent on SGML. Mission-critical, large-scale documentation is maintained in SGML by governments, aerospace, defense, automotive, telecommunications, and computing, to name only a few.

HTML: best-known SGML application

The most powerful information system in history, the World Wide Web, is driven by the best-known SGML application – HTML. Yet HTML, unlike other SGML applications, is used mainly for very short documents. And the fact that HTML is SGML is rarely known, even to the people using it.

XML: streamlined SGML subset for the WWW

The latest excitement surrounding SGML is over XML – a streamlined subset for the World Wide Web that isn't focused on document processing at all, large-scale or small. It is intended chiefly for data interchange. And once again, the fact that it is the same SGML that is used for documenting nuclear power plants and offshore oil rigs is hardly known.

How can SGML do such diverse things, on such a grand scale, and yet remain nearly invisible to the world at large? The answer isn't simple – if it were, SGML would be much better known and understood. For a first step toward that answer, we should examine the excitement over XML.

2 Why Is XML So Exciting?

Although XML and HTML are both based on SGML, there is a substantial difference between them. Consider how the markup for a comparative shopping guide for computers might look.

shopping guide in HTML

```
<p>P200 Laptop
<br>Friendly Computer Shop
<br>$1438
<p>P200 Laptop
<br>Discount Computer Warehouse
<br>$1299
```

shopping guide in XML

```
<product>
<model>P200 Laptop</model>
<dealer>Friendly Computer Shop</dealer>
<price>$1438</price></product>
<product><model>P200 Laptop</model>
<dealer>Discount Computer Warehouse</dealer>
<price>$1299</price>
</product>
```

Both of these may appear the same in your browser, but the XML data is smart data. HTML tells how the data should look, but XML tells you what it means. With XML, your browser knows there is a product, and it knows the model, dealer, and price. From a group of these it can show you the cheapest product or closest dealer without going back to the server.

Unlike HTML, with XML you create your own tags, so they describe exactly what you need to know. Because of that, your client-side applications can access data sources anywhere on the Web, in any format. New *middle-tier* application servers sit between the data sources and the client, translating everything into your own task-specific XML.

XML data is a smart document

But XML data isn't just smart data, it's also a smart document. That means when you display the information, the model name can be a different font from the dealer name, and the lowest price can be highlighted in red. Unlike HTML, where text is just text to be rendered in a uniform way, with XML text is smart, so it can control the rendition. And you don't have to decide whether your information is data or documents; in XML, it is always both at once. You can do data processing or document processing or both at the same time.

But XML can do even more. In a recent editorial, *InfoWorld* called XML "a major shift in computing technology". It said that XML could:

- End the browser wars: create once, distribute everywhere;
- Provide a data-neutral file format for Internet-commerce;
- Bridge transaction processing architecture models (Enterprise JavaBeans, Windows NT);
- Speed Web performance with more efficient data interchange.

To return to our question: How is it that one technology – SGML and its derivatives – can be used successfully for such a wide range of purposes? It is because SGML is not a processing technology; it is a document representation technology. SGML is unique among information representations for its ability to preserve the abstract data

content of arbitrarily complex information structures. To appreciate the implications of this fact, let us take a closer look at the way documents are represented in computers.

3 Document Representation

Document representation can be a complex issue, combining as it does the concepts of presentability, document notation, data object notation, and character sets. Even storage methods – such as files, databases, and document management systems – are involved, because they can change a representation in subtle or undocumented ways. The relationship of all these concepts is illustrated in Figure 1.

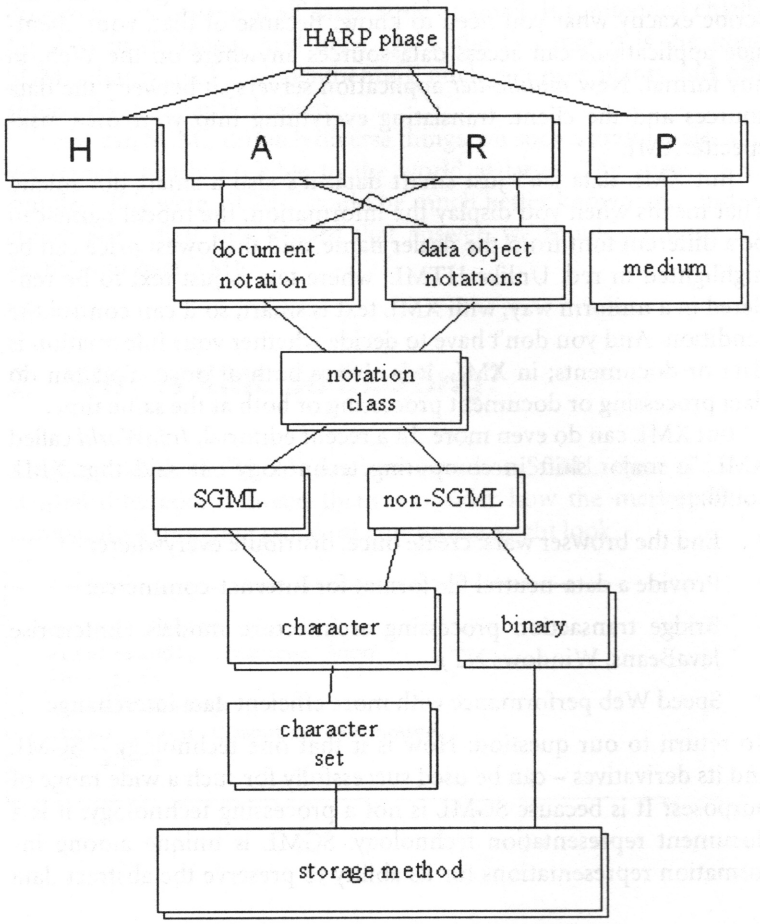


Figure 1 *Document Representation Components*

Of all the components of document representation, it is presentability that is the most important for understanding SGML and its future direction.

All of these document representation issues are discussed at length, with extensive tutorials, in my book, the *SGML Buyer's Guide* (Goldfarb/Pepper/Ensign 1997).

Presentability is a trait of a document that describes the document's readiness for presentation. A document can either be rendered (presentable) or abstract (unpresentable), in either the physical world or in the computer. The combinations yield four phases of presentability – Human thoughts, Abstraction, Rendition, and Presentation – derived as follows:

the four phases of presentability

	Physical world	Computer
Unrendered	(H)uman thoughts	(A)bstraction
Rendered	(P)resentation	(R)endition

We call presentability the "HARP" trait, because of the initials of the four phases (and because "presentability" is such a mouthful). It could also have been called the "usability" trait, for reasons that will be described.

The concept of presentability is perhaps best grasped by considering the flow of document information among humans, with and without computer assistance.

Without a computer: A human thinks, then presents a rendition of his thoughts in a physical medium – like voice or paper – so it can be perceived by other humans. A rendition has a "style" – a set of artistic characteristics. The style helps the rendition communicate to a specific audience in the most effective and appealing way.

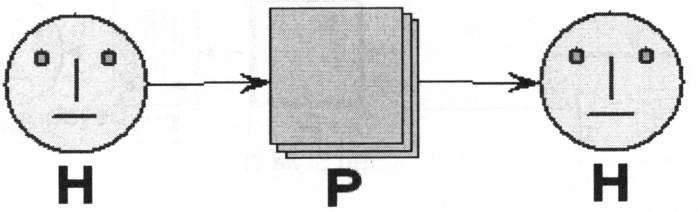


Figure 2
Information Flow without a Computer

With a computer and a word processor: A human thinks, then captures a rendition of his thoughts in computer storage. The rendition can later be presented in a physical medium – like a computer screen or printout – so it can be perceived by humans.

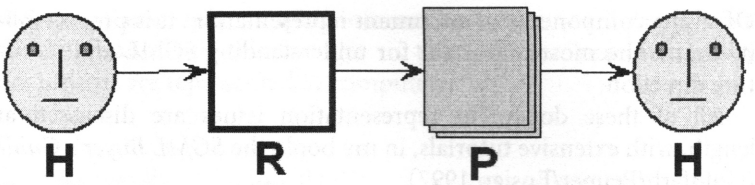


Figure 3
Information Flow with a Computer and a Word Processor

With a computer and SGML: A human's thoughts are captured as abstract thoughts in computer storage. The abstract thoughts can be rendered many times within the computer, each rendition in a different style for a different audience. The multiple renditions can be presented in different physical media, so they can be perceived by humans under a variety of conditions.

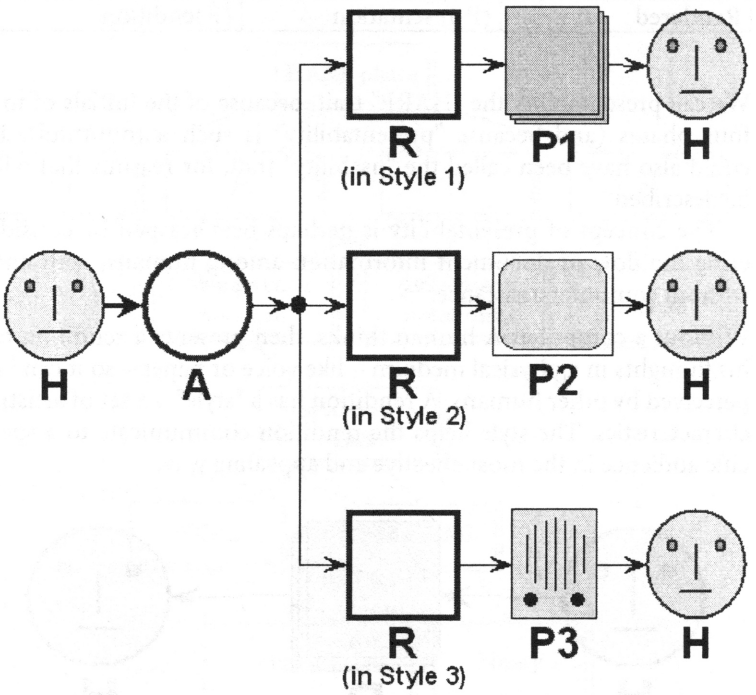


Figure 4
Information Flow with a Computer and SGML

The vital point about presentability is that there are more potential uses for an unpresentable (that is, abstract) document than for a presentable (that is, rendered) one. That makes the abstract phase the best choice for a document archive with long life expectancy and lots of expected reuse. Lord Chesterfield said "Style is the dress of thoughts", and naked abstractions can always be dressed as needed to

fit the situation. On the other hand, there may be greater value (or at least, greater expense) associated with a rendition, especially if it was created by an interactive process.

The complement of presentability, then, is usability. A document in the abstract phase is maximally reusable, since multiple renditions and alternative abstractions can be created from it. However, only a rendition is ready to be presented to a human.

a rendition has style; an abstraction does not

In summary: Thoughts are abstractions that exist in the real world, in a human mind. Presentations are renditions that exist in the real world, where humans can perceive them. A rendition has style; an abstraction does not.

Abstractions and renditions can also exist as data files in a computer. It is possible to create multiple renditions of the same abstraction by applying different styles. SGML experts frequently call abstractions "structured information". That is technically accurate, but insufficient to distinguish abstractions from renditions, which also have structure.

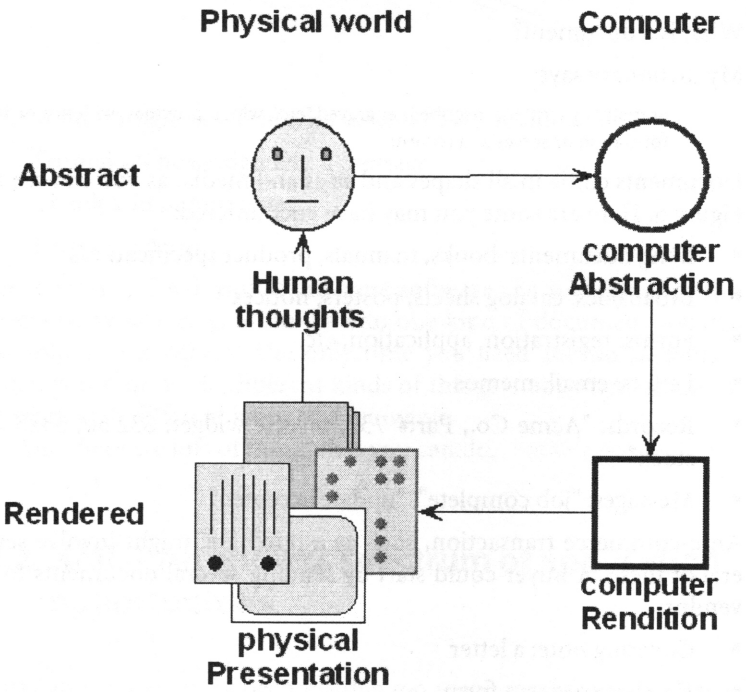


Figure 5
Summary of HARP Phases

4 The Role of XML

The power of SGML is that it can represent abstractions of any structural complexity; anything, for example, that can be represented in a relational, or object-oriented, or hybrid database. But SGML can do so without relying on proprietary database formats or software.

XML retains the representational power of SGML

XML retains this power, but with much less parsing overhead. That is because it eliminates almost all of SGML's syntactic customization and options, while retaining the ability to create your own document types, element types, attributes, and hyperlinks.

XML scales down so well that it can be used efficiently for the smallest pieces of data. And it has the added benefit of enabling that data to be processed both as data and as a document.

4.1 Is XML for documents or for data?

What is a document?

My dictionary says:

> Something written, inscribed, engraved, etc., which provides evidence or information or serves as a record:

Documents come in all shapes and sizes and media, as you can see in Figure 6. Here are some you may have encountered:

- Long documents: books, manuals, product specifications
- Broadsides: catalog sheets, posters, notices
- Forms: registration, application, etc.
- Letters: email, memos
- Records: "Acme Co., Part# 732, reverse widget, $32.50, 5323 in stock"
- Messages: "job complete", "update accepted"

An e-commerce transaction, such as a purchase, might involve several of these. A buyer could start by sending several documents to a vendor:

- Covering note: a letter
- Purchase order: a form
- Attached product specification: a long document

Figure 6
Documents in Different Shapes, Sizes, and Media

The vendor might respond with several more documents:

- Formal acknowledgment: a message
- Thank you note: a letter
- Invoice: a form

the beauty of XML ...

The beauty of XML is that the same software can process all of this diversity. Whatever you can do with one kind of document you can do with all the others. The only time you need additional tools is when you want to do different kinds of things – not when you want to work with different kinds of documents.

And there are lots of things that you can do.

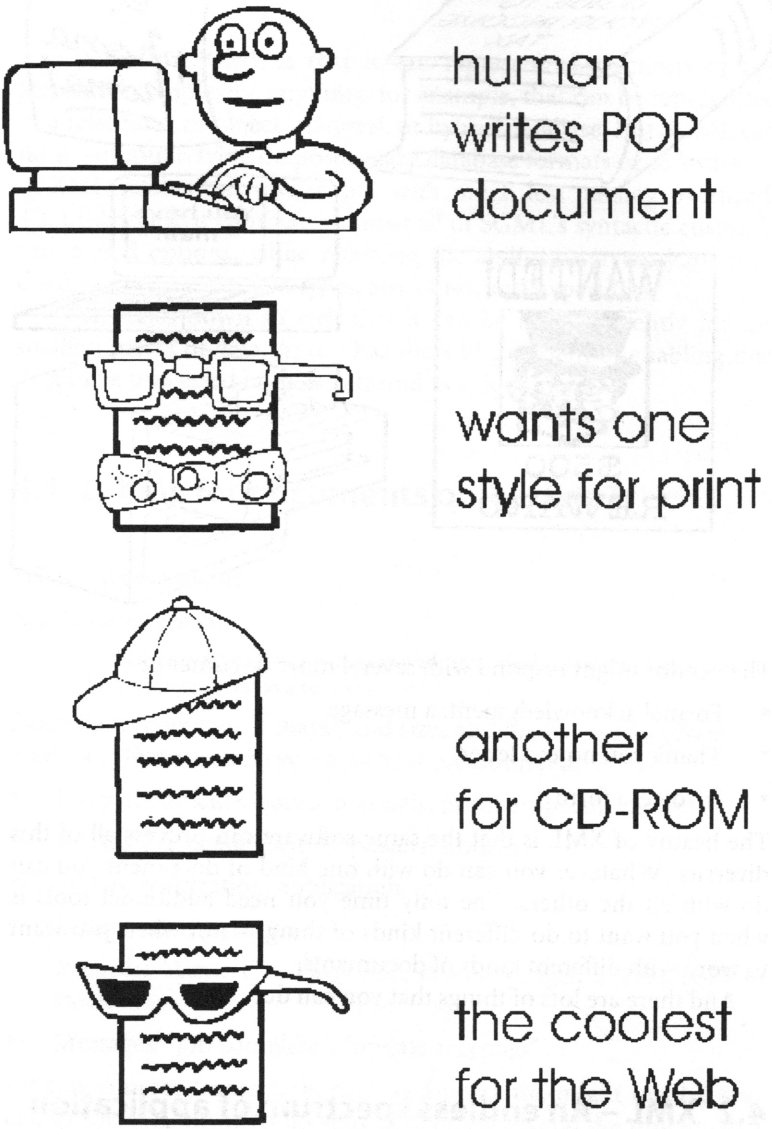
4.2 XML – An endless spectrum of application opportunities

Sorry about that, we've been reading too many marketing brochures. But it's true, nevertheless.

At one end of the spectrum we have the grand old man of generalized markup: POP – Presentation-Oriented Publishing.

You can see him in Figure 7.

Figure 7
POP Application



At the other end of the spectrum is that darling of the data processors, MOM – Message-Oriented Middleware. She smiles radiantly from Figure 8.

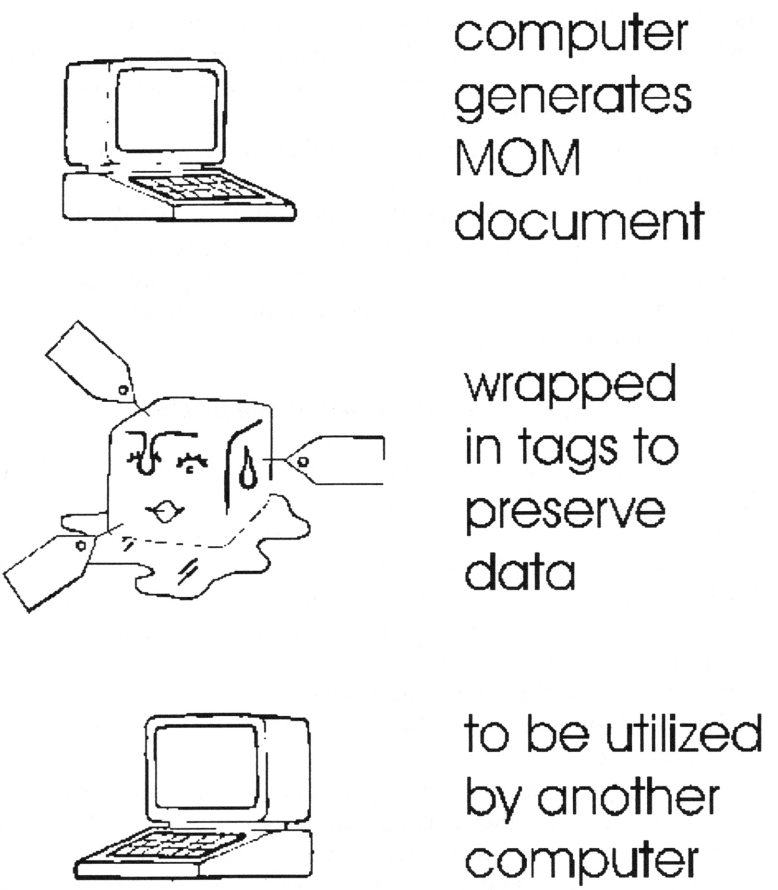


Figure 8
MOM Application

Let's take a closer look at both of them.

4.2.1 Presentation-oriented publishing (POP)

POP was the original killer app for SGML

POP was the original killer app for SGML, XML's parent, because it saves so much money for enterprises with Web-sized document collections.

POP documents are chiefly written by humans for other humans to read.

Instead of creating formatted renditions, as in word processors or desktop publishing programs, XML POP users create unformatted abstractions. That means the document file captures what is in the document, but not how it is supposed to look.

To get the desired look, the POP user creates a stylesheet, a set of commands that tell a program how to format (and/or otherwise process) the document. The power of XML in this regard is that you

don't need to choose just one look – you can have a separate stylesheet for every purpose. We know that all office suites have some degree of stylesheet support today, but the SGML languages did it first, and are still the only way to do it cleanly. At a minimum, you might want one for print, one for CD-ROM, and another for a Web site.

POP documents tend to be long-lived

POP documents tend to be (but needn't be) long-lived, large, and with complex structures. When delivered in electronic media, they may be interactive. How they will be rendered is of great importance, but, because XML is used, the rendition information can be – and is – kept distinct from the abstract data.

4.2.2 Message-oriented middleware (MOM)

MOM is the killer app for XML on the Web

MOM is the killer app – actually, a technology that drives lots of killer apps – for XML on the Web.

Middleware, as you might suspect from the name, is software that comes between two other programs. It acts like your interpreter or guide might if you were to visit someplace where you couldn't speak the language and had no idea of the local customs. It talks in the native tongue, using the native customs, and translates the native replies – the messages – into your language.

MOM documents are chiefly generated by programs for other programs to read.

Instead of writing specialized programs (clients) to access particular databases or other data sources (servers), XML MOM users break the old two-tier client/server model. They introduce a third tier, the *middle tier*, that acts as a data integrator. The middle-tier application server does all the talking to the data sources and sends their messages in XML to the client.

That means the client can read data from anywhere, but only has to understand data that is in XML documents. The XML markup provides information about the data (i.e., metadata) that was in the original data source schema, like the database table name and field names (also called *cell* or *column* names).

MOM users typically don't care much about rendition

The MOM user typically doesn't care much about rendition. He does care, though, about extracting the original data accurately and making use of the metadata. His client software, instead of having a specialized module for each data source, has a single "XML parser" module. The parser is the program that separates the markup from the data, just as it does in POP applications.

And just like POP applications, there can be a stylesheet, a set of commands that tell a program how to process the document. It may not look much like a POP stylesheet – it might look more like a

script or program – but it performs the same function. And, as with POP stylesheets, there can be different MOM stylesheets for different document types, or to do different things with message documents of a single document type.

extra benefit to XML three-tier MOM applications

There is an extra benefit to XML three-tier MOM applications in a networked environment. For many applications, the middle-tier server can collect all of the relevant data at once and send it in a single document to the client. Further querying, sorting, and other processing can then take place solely on the client system. That not only cuts down Web traffic and overhead, as *InfoWorld* observed, but it vastly improves the end-user's perceived performance and his satisfaction with the experience.

MOM documents tend to be (but needn't be) short-lived, non-interactive, small, and with simple structures.

4.2.3 Opposites are attracted

To XML, that is!

How is it that XML can be optimal for two such apparently extreme opposites as MOM and POP? The answer is, the two are not really different where it counts.

In both cases, we start with abstract information. For POP, it comes from a human author's head. For MOM, it comes from a database. But either way, the abstract data is marked up with tags and becomes a document.

XML documents are special

But XML documents are special. An application can do three kinds of processing with one:

- *Parse it*, in order to extract the original data. This can be done without information loss because XML represents both metadata and data, and it lets you keep the abstractions distinct from rendition information.
- *Render it*, so it can be presented in a physical medium that a human can perceive. It can be rendered in many different ways, for delivery in multiple media such as screen displays, print, Braille, spoken word, and so on.
- *Hack it*, meaning "process it as plain text without parsing". Hacking might involve cutting and pasting into other XML documents, or scanning the markup to get some information from it without doing a real parse.

The real revelation here is that data and documents aren't opposites. Far from it – they are actually two states of the same information.

The real difference between the two is that when data is in a database, the metadata about its structure and meaning (the schema) is

stored according to the proprietary architecture of the database. When the data becomes a document, the metadata is stored as markup.

notation rules needed for mixture of markup and data

A mixture of markup and data must be governed by the rules of some notation. XML and SGML are notations, as are RTF and Word file format. The rules of the notation determine how a parser will interpret the document text to separate the data from the markup.

Take another look at Figure 1.

Notations are not just for complete documents. There are also data object notations, such as GIF, TIFF, and EPS, that are used to represent such things as graphics, video (e.g., MPEG), and audio (e.g., AVI). Document notations usually allow their documents to contain data objects, such as pictures, that are in the objects' own data object notations.

Data object notations are usually (not always) in binary; that is, they are built-up from low-level ones and zeros. Document notations, however, are frequently character-based. XML is character-based, which is why it can be hacked.

In fact, a design objective of XML was to support the "desperate Perl hacker" – someone who needs to write a program in a hurry, using a scripting language like Perl, and who doesn't use a real XML parser. Instead, his program scans the XML document as though it were plain text. The program might search for markup strings, but can also search for data.

opportunities for synergy

Since databases and documents are really the same, and MOM and POP applications both use XML documents, there are lots of opportunities for synergy.

4.2.4 MOM and POP – They're so great together!

Classically, MOM and POP were radically different kinds of applications, each doing things its own way with different technologies and mental models. But POP applications frequently need to include database data in their document content – think of an automotive maintenance manual that has to get the accurate part numbers from a database.

Similarly, MOM applications need to include human-written components. When the dealer asks for price and availability of the automotive parts you need, the display might include a description as well.

generalized markup allowing MOM and POP to go together

With the advent of generalized markup, the barriers to doing MOM-like things in POP applications began to disappear. Some of the POP-like SGML applications that have been in use for many years appear to have invented the middle tier on their own. And

now, with the advent of XML, MOM applications can easily incorporate POP functionality as well.

no more a difference in kind between MOM and POP

In fact, we'd go so far as to say there is no longer a difference in kind between the two, only a difference in degree. There really is "an endless spectrum of application opportunities". It is a multi-dimensional spectrum where applications need not be implemented differently just because they process different document types. The real differentiators are other document characteristics, like persistency, size, interactivity, structural complexity, percentage of human-written content, and the importance of eventual presentation to humans.

At the extremes, some applications may call for specialized (or optimized) techniques, but the broad central universe of applications can all be implemented similarly. Much of the knowledge that POP application developers have acquired over the years is now applicable to MOM applications, and vice versa.

You can find detailed descriptions of dozens of XML applications – MOM, POP, and hybrid – in my book, *The XML Handbook* (Goldfarb/Prescod 1998).

5 Revising the SGML Standard

The original clients for SGML were large enterprises using POP applications in controlled and robust mainframe environments. In contrast, the new users of XML are predominantly using MOM applications in open and relatively unreliable networked environments.

a need for changes to ISO 8879

As you might expect, these new uses created a need for changes to ISO 8879 in order for XML to be a proper subset of SGML. Fortunately, a revision process was already under way when XML arrived on the scene in 1996.

SGML and its related standards have been developed in the *International Organization for Standardization (ISO)* under the same group of leaders for the past twenty years. During that time, the development committee has changed names several times. Most recently it was promoted to a full-fledged technical subcommittee: ISO/IEC JTC1/SC34. For clarity, I will refer to it only by that name.

5.1 Revision History and Policy

In May, 1990, in the light of five years of user experience with SGML, SC34 directed me, as Project Editor, to lead a complete review of ISO 8879. The idea was to see whether user requirements or technology had changed sufficiently to justify revisions to the standard.

guaranteeing backward compatibility

We developed a policy document, N1289: Future development of ISO 8879, that would protect existing users of SGML by guaranteeing backward compatibility. It says:

> Any document that is a conforming SGML document according to the current standard shall continue to be a conforming document under the provisions of future versions of the standard.

5.2 SGML Extended Facilities

We took our time with the review, for several reasons. One was the importance of building user confidence in the standard by maintaining its stability. Another was the desire to be thorough.

A third reason was the realization that we lacked something essential. We had no adequate formal definition of the output of an SGML parser: the structure and other properties of a document that are described by the SGML language.

a specification for property sets and groves

Our response to that problem was to develop a specification for property sets and groves in general.

Groves are collections of trees, the structure of a parsed document, and the SGML property set in particular. The latter became the basis for navigation and querying in DSSSL and in HyTime, and has influenced the XML DOM and XLink work.

Both specifications are currently published in the *SGML Extended Facilities,* Annex A of ISO/IEC 10744 HyTime 2d Edition. SC34 has voted to move the extended facilities to the SGML standard when it is republished.

Annex A also includes, among other things, definition requirements for:

- Lexical Types (LTDR)
- Architectural Forms (AFDR)
- Formal System Identifiers (FSIDR)

A document can use any or all of these facilities modularly.

For a reader's guide and the text of the extended facilities, see (HyTime Users' Group 1998).

5.3 Extended Naming Rules

The first change in nine years was made to the SGML standard itself at the request of the Japanese delegation to SC34. In 1997, SC34 approved Annex J to ISO 8879, the *Extended Naming Rules.*

Annex J allows changes to the SGML declaration that make it easier to specify large character sets. It also makes it easier to specify name characters for languages that have no concept of upper- and lower-case. These changes enhance the usability of SGML with Asian languages.

5.4 WebSGML Adaptations

priority to XML user requirements

SC34 originally intended to produce the revised version of SGML all at once. However, the XML development activity presented clear user requirements for certain changes, and we decided to give them priority. Accordingly, on December 5, 1997, SC34 approved the *WebSGML Adaptations*, consisting of new Annexes K and L to ISO 8879.

WebSGML offers many new capabilities, including some that SGML users have requested for a long time. Some of the favorites include:

- SGML declaration reference
- White space handling: KEEPALL
- Optional quantities and capacities
- Hex character references
- Empty elements can have end-tags
- SHORTTAG unbundled
- Multiple and universal ATTLIST
- Duplicate nmtoken values permitted
- Domain names as public text owners
- Element type declaration

There are too many even to mention them all here, let alone supply all the details, but you can read about them at my Web site.

As a result of WebSGML, all XML documents – whether valid or merely well-formed – are also conforming SGML documents.

5.5 DTD-less Parsing

One of the original objectives for SGML was the ability to validate a document when editing, then to process the document without validating it again. That is why ISO 8879 conformance allows both validating and non-validating systems. It was intended that a non-validating system would not have to access a DTD.

SGML: type-validity required

Unfortunately, some aspects of markup minimization prevented DTD-less parsing, but these have now been remedied. Prior to WebSGML, a conforming SGML document required an explicit DTD association, and the instance had to conform to the DTD. Such a document is said to be *type-valid.*

WebSGML: type-validity optional

In WebSGML, type-validity is optional. In this respect, it is similar to XML. But WebSGML goes a step further in allowing partial type-validity: it is permissible for DTD declarations to be omitted, but the instance must still conform to any DTD declarations that are present.

Also, XML requires all document instances to be what WebSGML calls *fully-tagged.* That is, there must be a start- and end-tag for each element, and attributes cannot be defaulted.

XML's so-called *empty-element tag* conforms to this rule. In SGML terms it is actually a start-tag followed by a null end-tag.

For DTD-less parsing, WebSGML also allows markup minimization that does not require access to the DTD. For example, the end-tag of an element can end any open children of that element.

5.6 Entity Constraints

Another innovation in WebSGML is the ability to constrain the use of entity references. They can be prohibited, limited to internal entities, or used freely as has always been permitted in SGML.

A further possibility – one that is used by XML – is to require that entities be *integrally-stored.* That is, elements and markup strings must end in the same entity in which they begin.

5.7 Datatypes

A facility that is new to WebSGML – and not yet available in XML – is strong datatyping for attributes. You can use any form of modeling

notation, such as regular expressions, to define a datatype for an attribute value.

In the following example, `regexp` is defined as a notation. It requires a `model` and a `max` (maximum length) to be specified when expressing a datatype. The date attribute of chapter is defined using the `regexp` notation. The model is "[0-9]+-[0-9]+]" and the maximum length of a value is 7 characters.

```
<!NOTATION regexp SYSTEM "url of regexp handler">
<!ATTLIST #NOTATION regexp
  model CDATA #IMPLIED
  max NMTOKEN "128">

<!ELEMENT chapter (%chapter-stuff;)>
<!ATTLIST chapter
  date DATA regexp [model="[0-9]+-[0-9]+]"
max="7"] #IMPLIED>

<chapter date="12-97">
```

When the date attribute is parsed, the value is passed to the `regexp` handler identified in the notation declaration, along with the `model` and `max` parameters from the definition of `date`. The `regexp` handler can then tell the parser whether the date value conforms to the datatype.

5.8 DTD Data Entities (DTD Schemas)

A DTD performs the same role in SGML and XML as a schema does in a database management system (DBMS). Originally, SGML only allowed DTDs to be expressed using markup declarations. Nevertheless, many SGML products also used optimized proprietary notations for DTDs.

using XML documents to express DTD schemas

And recently, several proposals have been made for using XML documents to express DTD schemas. These schemas would express not only the standardized DTD properties that can be expressed by markup declarations, but additional ones as well (such as inheritance, synonyms, etc.).

In response to these requirements, SC34 is in the process of finalizing an enhancement to WebSGML that would allow alternative notations for DTD schemas. From the standpoint of SGML, they would be safe to use under the same conditions in which WebSGML allows the DTD to be ignored.

Here is how you might declare that such a notation is used in the external subset of the document type declaration:

```
<!DOCTYPE mydoc SYSTEM "location of external subset"
        CDATA some-schema-notation [
<!NOTATION some-schema-notation
       SYSTEM "location of schema interpreter>
]>
```

The *schema interpreter* must communicate the standardized DTD properties to the SGML parser, which will use them as if they had been expressed by markup declarations.

Similar declarations are possible for external parameter entities, and different DTD schemas can be used in each case.

5.9 Completing the Revision

A number of activities remain before a complete revised edition of ISO 8879 can be published. There are many corrections, clarifications, and minor design changes, most of which have been documented in the reports of meetings of SC34 (see website: SGMLsource).

development of modularized DTDs

One of the most interesting design efforts is the development of modularized DTDs. Modules would allow a robust method of combining components of multiple DTDs with duplicate object names, while preserving the context of the original namespaces. The original proposal was made by the Japanese delegation to SC34, and Japan continues to lead the development effort.

Also under consideration is the possibility of clarifying the presentation of the standard by replacing much of the normative text with formal notations.

6 The Future of SGML

the tail that wagged the dog

When the World Wide Web became a phenomenon, just five short years ago, HTML suddenly became the tail that wagged the dog. A vastly larger number of people knew about it and used it than had ever used its parent, SGML. And SGML users found themselves in

the position of having to convert their documents from their own DTDs to HTML in order to display them in Web browsers.

Lots of SGML users disliked HTML because it was rendition-oriented, but some experts embraced it. We heard speakers at GCA conferences claiming that "HTML is all the markup we'll ever need."

Well, it isn't, and we eventually found that out. In fact, XML is, in part, the Web community's response to the shortcomings of HTML.

XML and SGML don't even compete

Now we are hearing the same things about XML that we heard about HTML. Only now they are much louder, they have much more support from the SGML community, and they are much more justified – but still just as wrong. As with HTML, there are two sides to the argument.

Some claim that XML will replace SGML because there will be so much free and low-cost software. Others assert that XML users, like HTML users before them, will discover that they need more of SGML and will eventually migrate to the full standard.

Both assertions are nonsense ... XML and SGML don't even compete.

XML: simplified subset of SGML optimized for the Web

XML is a simplified subset of SGML. The subsetting was optimized for the Web environment, which implies data-processing-oriented (rather than publishing-oriented), short life-span (in fact, usually dynamically-generated) information. The vast majority of XML documents will be created by computer programs and processed by other programs, then destroyed. Humans will never see them.

Eliot Kimber, a member of both the XML and SGML standards committees, says:

for certain use domains XML is simply not sufficient

> There are certain use domains for which XML is simply not sufficient and where you need the additional features of SGML. These applications tend to be very large scale and of long term; e.g., aircraft maintenance information, government regulations, power plant documentation, etc.
>
> Any one of them might involve a larger volume of information than the entire use of XML on the Web. A single model of commercial aircraft, for example, requires some four million unique pages of documentation that must be revised and republished quarterly. Multiply that by the number of models produced by companies like Airbus and Boeing and you get a feel for the scale involved.

I agree wholeheartedly with Eliot.

I invented SGML, I'm proud of it, and I'm awed that such a staggering volume of the world's mission-critical information is represented in it.

I'm also proud of HTML. I'm delighted that Tim Berners-Lee created it and used it in the World Wide Web. And I'm gratified that he based so much of it on the original element types that Ed Mosher and I created for GML – SGML's precursor – over 25 years ago.

And I'm thrilled with XML. I'm proud of my friend Jon Bosak who made it happen, and I'm excited that the World Wide Web is becoming XML-based.

Consider the two "handbooks" that I've written. *The SGML Handbook* (Goldfarb 1990) has been in print since 1990 and continues to sell well. But *The XML Handbook* (Goldfarb/Prescod 1998) sold more copies in its first eight weeks than *The SGML Handbook* had sold in eight years!

the world needs SGML and HTML and XML

Clearly, the world needs SGML and HTML and XML.

The major producers of documents will always need SGML – nothing else can do the job. It provides the enormous flexibility needed. It is stable in a way that nothing but an honestly ISO-developed International Standard can be. And any other representation – such as HTML or XML – can be generated from SGML automatically.

So the SGML standard will continue to respond to the broad demanding needs of the large user, and will continue to be a source from which XML can draw future enhancements. But it will carry on doing so in its current relative obscurity.

As with HTML, the public attention and excitement will belong to the child, not the parent. XML will have an order of magnitude more users, who will revolutionize the world's business and financial relations by exchanging commercial data over the World Wide Web.

But the unimaginably large SGML repository of long-lived intellectual property will continue to grow. There will always be vastly more persistent information in SGML than in HTML and XML combined.

Acknowledgments

I am grateful to Fujitsu Network Communications for supporting the development of this paper. I would also like to thank Chet Ensign, Steve Pepper, and Paul Prescod, the co-authors with whom I wrote the books from which parts of this paper are derived.

References

(Goldfarb 1990)
Goldfarb, Charles F.: The SGML Handbook. Oxford: Oxford University Press, 1990

(Goldfarb et al. 1997)
Goldfarb, Charles F.; Pepper, Steve; Ensign, Chet: SGML Buyer's Guide. Upper Saddle River, N.J.: Prentice Hall, 1997

(Goldfarb/Prescod 1998)
Goldfarb, Charles F.; Prescod, Paul: The XML Handbook. Upper Saddle River, N.J.: Prentice Hall, 1998

(HyTime Users' Group 1998)
HyTime Users' Group: HyTime Users' Group Home Page. o.O., September 1998. Information on the SGML Extended Facilities and HyTime. Available from Internet: http://www.hytime.org

(SGML Source Home Page 1997)
Charles F. Goldfarb's SGML Source Home Page. o.O., December 1997. Information on the SGML revision. Available from Internet: http://www.SGMLsource.com